Deep Learning Assignment 2

Murad Bozik
s2619822

Yuchi Zhang
s2724952

Hansha Leng
s2604825

April 26, 2020

1 Introduction

In this assignment, we learned tensorflow and keras frameworks. We gain experience by building multilayer perceptron (MLP) and convolutional neural network (CNN) for MNIST and Fashion MNIST datasets. Task 1 and 2 includes our experiments and findings with MLP and CNN. In task 3, we used our knowledge to built a deep neural network (DNN) to be able to predict time from analog clock images. Our findings and short description of each task written into the section for each tasks.

2 Preliminaries

In this section, we describe the approach and tools we use in this report. In order to build neural networks we used keras library with tensorflow backend. We tried to optimize our models using grid and randomized cross-validation searches from scikit-learn library.

2.1 General Approach

We divided task 1 into two files *task1_mlp.py* and *task1_cnn.py*. In each file, we tested models and tried to get best results for Fashion MNIST dataset. We prefered Fashion MNIST dataset for final evaluation if our models, since the performances of models was lower on this dataset. This allowed us to see improvements and effects of tuning parmeters much more clearly.

We started with the plain structure for both models as described in Gèron's textbook[1]. As first step, we kept the model structure the same as in the book. We changed parameters to see their effects on improvements. These parameters presented in table 1. Then, we tuned best parameters according to our observations from first step and changed the structure of models to get better results. We decided our best models after second step. And saved these models to use in task 2.

In task 2, we permuted MNIST and Fashion MNIST datasets and retrained our best models with permuted datasets. We evaluated our models on test sets. Our findings regarding this part presented in section 4.

In task 3, we built a network which can tell time from an analog clock image.

Step	Parameter	Values
1	Initializers	Zeros, Ones, RandomNormal
	Activation functions	sigmoid, relu, tanh
	Optimizers	SGD, RMSprop, Adagrad, Adam
	Regularizers	l1(0.1), l2(0.1), l1_l2(0.01, 0.01)
2	Number of Hidden Dense Layers	1, 2, 3
	Number of Neurons	arange(1,100)
	Number of Hidden Convolution Layers	1, 2, 3
	Initial filter size for Conv Layers	16, 32, 64
	Learning Rate	reciprocal(3e-4, 3e-2)

Table 1: Tuned Parameters

Model (Training Epochs)	MNIST			Fashion MNIST		
	Train	Valid	Test	Train	Valid	Test
MLP (30 epochs)	99.37	98	97.68	93.05	89	85.54
CNN (10 epochs)	98.27	98.94	98.94	89.86	90.96	89.73

Table 2: Performances of Models

3 Task 1: Learning basics of Keras and TensorFlow.

3.1 MLP model

The MLP model described in Géron's textbook has 2 hidden layers with 300 and 100 neurons. Output layer has 10 neurons, because both datasets has 10 class to predict. We trained and evaluated MLP model on both datasets with 30 epochs. While MLP model exceeds 90% accuracy in second epoch on MNIST dataset, it could reached same level accuracy on 13th epoch on Fashion MNIST. Models' performances are demonstrated on Table 2.

In order to tune parameters, we created a function and used a wrapper for this function. In this way we could run GridSearchCV function from scikit-learn library. First, we investigated initializers. Although there are many other options, we limited our search with selected values not only for initializer, also for other parameters. Because, Grid search is an exhausting search whose computational cost increases rapidly with the numbers of parameters. The experiments we conduct showed that when the number of parameters is high, it cause a memory error during search.

The results of initizalizers showed us the important parameter for initializers is kernel initializer. RandomNormal is clearly the best initializer for kernel. However, for bias all initializers performed almost same results.

When it comes to activation functions, *tanh* and *relu* gives better results than sigmoid function. Their scores are around 91% but sigmoid function gives 30% score.

For optimizers, we prefered to choose same learning rate for all optimizers to make it fair for comparison. RMSprop and Adam optimizers provided 97% mean test score. Adagrad gave 90%. But, schothastic gradient decent (sgd) performed 50% mean score. Which was surprising result, because in the Géron's book. MLP was introduced with SGD optimizer.

None of regularizers provided more than 11% score. Then, we repeated this experiment by adding *None* option into parameter distribution. But, the results didn't changed. That's why, we preferred not to use regularizers for the rest of experiments.

In step 2, we chose the best parameters so far. With the help of RandomizedSearchCv function, we calculated the best structure for MLP model. It performed 85% on Fashion MNIST test set. The best parameters for MLP model are shown in Table 3.

Step	Parameter	Best Values
1	Initializers	bias: Zeros kernel: RandomNormal
	Activation functions	relu
	Optimizers	Adam
	Regularizers	l1_l2(0.01, 0.01)
2	Number of Hidden Layers	1
	Number of Neurons	99
	Learning Rate	0.00042138860510843035

Table 3: Best Parameters for MLP model

3.2 CNN Model

CNN model uses convolution layers(Conv2D) to apply convolution and pooling operation(MaxPooling2D) to reduce dimensions. In the basic CNN model described in the book, it has 4 hidden conv layers, each two Conv2D layers are followed by MaxPooling2D layer. After the Flatten layer it has the same structure with MLP model, except in order to prevent overfitting dropout has been applied.

CNN model works quite well with images. It needs an extra dimension for color channel. That's why, for fitting, evaluating, predicting operations X data should be preprocessed. Evaluation of CNN

model has been made with only 10 epochs. Although we train CNN model $\frac{1}{3}$ factor of MLP epochs, CNN provides much better results. The scores can be seen in Table 2. This shows that CNN model is more capable of dealing with complex features of images. In each layer, it analyses the features of an image. That's why it provided better results for both datasets.

We used the same methodology to evaluate effects of parameters. We got the similar results for initializers. Top 3 ranking belongs to RandomNormal kernel initializers. It means kernel initializer is more important and has effect on result. This results are predictable, because the model has more than 1 million weights for tuning. This is normal. When the random values are used to initalize weights they can converge quicker than other initializers.

The effects of activation functions are also similar with MLP. Sigmoid function is the worst one. This time tanh function gave us better results. Adam Optimzer gives the best result among the four optimizer we choose. For regularizers, although we added None parameter into search parameters, all the results are too similar. The difference between the first rank score and the one we did not apply regularizer is just 1%. For this reason, for the next step we preferred not to use regularizers.

In step 2, we preferred to keep general structure same which means the model builder function creates double Conv2D layers followed by MaxPooling2D layer. And each Conv2D layer pair doubling the filter size. The rest of the function is same with MLP model builder. We used randomized cross validation search. The results are presented in Table 4.

Step	**Parameter**	**Best Values**
1	Initializers	bias: Zeros kernel: RandomNormal
	Activation functions	tanh
	Optimizers	Adam
	Regularizers	l1_l2(0.01, 0.01)
2	Initial Filter Size	32
	Number of Hidden Conv Layers	1
	Number of Hidden Dense Layers	1
	Number of Neurons	90
	Learning Rate	0.00150603481780574

Table 4: Best Parameters for CNN model

4 Task 2: The impact of obfuscating data

In this task, We permuted both datasets' training examples with the same random state. Permutation process in here is just swapping indexes randomly within the same image. How the images look like before and after permutation are shown in Figure 1 and Figure 2.

Figure 1: Permutation effect on MNIST sample

Models	Permuted MNIST			Permuted Fashion MNIST		
	Train	Valid	Test	Train	Valid	Test
MLP	99.9	97.36	96.89	90.81	88.20	87.03
CNN	95.09	96.02	91.43	84.92	86.76	85.23

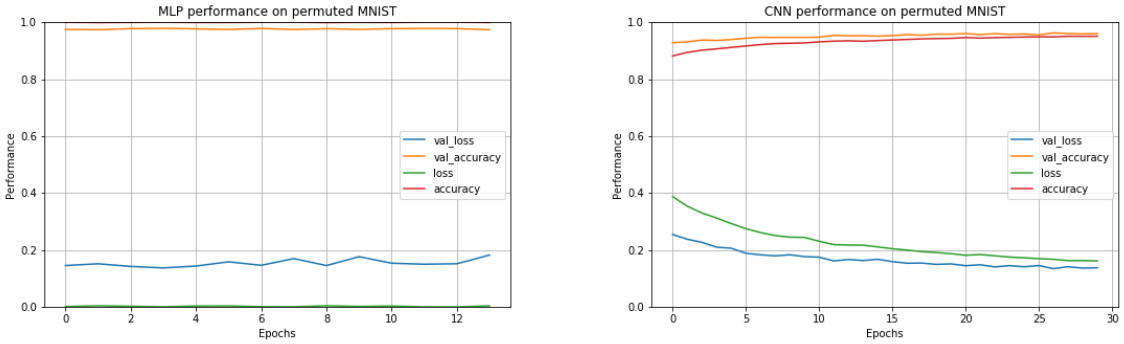
Table 5: Performances of Models on Permuted Datasets

Figure 2: Permutation effect on Fashion MNIST sample

After permutation, we loaded the best models of MLP and CNN from task 1. We retrained and evaluated these permuted datasets. The results of evaluations presented in Table 5.

4.1 Permuted MNIST

We performed trainings with 30 epochs, but in order to prevent overfitting we used early stopping callback from keras library. This callback stops when there is no further improvement on model. MLP model performed extremely good on MNIST dataset. Training completed in 14 epochs. the least accuracy is above 99%. CNN model couldn't reach this level although it completed all 30 epochs. The accuracies and losses are shown in Figure 3.



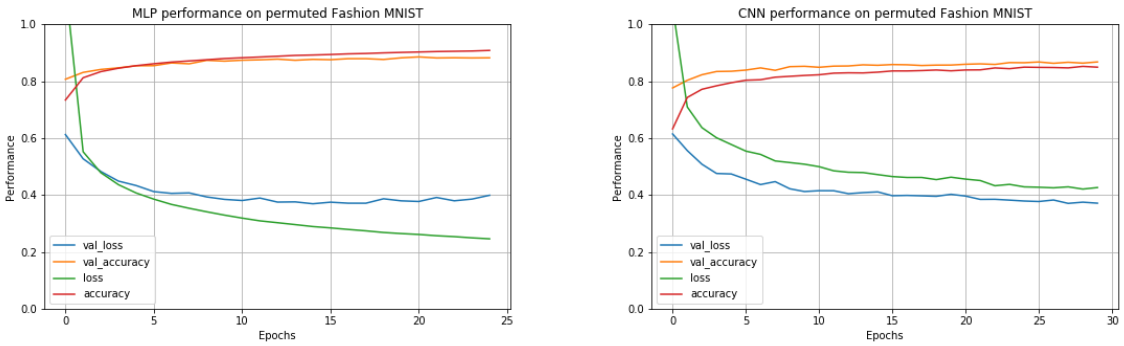
(a) MLP

(b) CNN

Figure 3: Accuracies and Losses on permuted MNIST

4.2 Permuted Fashion MNIST

The same evaluation has been made on permuted Fashion MNIST dataset. MLP could complete the training in 25 epochs, while CNN performed all epochs. MLP model gave better results again on this datset. Results are presented in Table 5. Figure 4 shows losses and accuracies graphs.



(a) MLP (b) CNN

Figure 4: Accuracies and Losses on permuted Fashion MNIST

5 Task 3: Developing a "Tell-the-time" network

5.1 Introduction

In this part, We implemented a CNN network. Convnet is used to identify images by converting the original image through layers into class fractions. The inspiration for CNN came from the visual cortex. Every time we see something, a series of neurons are activated, and each layer detects a set of features, like lines, edges. The higher levels will detect more complex features to identify what we are seeing. Deep learning CNN model is trained and tested. Each input image will be classified by a series of convolution layers with filters (Kernals), Pooling, and full connection layer (FC), and the objects with probability values between 0 and 1 will be classified by Softmax function. The following is the complete flow of CNN processing input images and classifying objects by value.

5.2 Design and Implementation

We use Keras to implement a CNN model. The figure shows as Figure 5. We have a 2d layer, a pooling layer, a flatten layer and a dens_1 layer and a output layer.Each Keras model is either built using the Sequential class (which represents a linear stack of layers) or the functional model class (which is more customizable). We will use a simpler sequence model because our CNN will be a linear layer stack.

5.3 Optimization and Error Decrease

In the Input layer (training data) the input volume is an image with the following dimensions: [width x height x depth (1)]. It's a matrix of pixel values. Convolution is the first layer to extract features from the input image, while Conv layer aims to extract features from the input data. Convolution preserves the relationship between pixels by using small squares of input data to learn the features of the image. The Stride is the number of pixel shifts on the input matrix. When the step length is 1, we move the filter to 1 pixel. When the step width is 2, we move the filter to 2 pixels at a time, and so on. The image below shows that the convolution will work with a step width of 2. Sometimes filters are not suitable for input images. We have two options: 1. Fill the picture with zero to make it fit; 2. Remove the parts of the image that are not suitable for the filter. This is called effective padding, which preserves only the valid part of the image. ReLU represents the rectifying linear element and is used for nonlinear operations. The output is (x) = Max (0,x). Why ReLU is important: the purpose of ReLU is to introduce nonlinearity into our ConvNet. Because the real world data wants us to ConvNet learn nonnegative linear values.

5.4 Evaluation

Figure 6 shows the Runtime screenshot of our CNN in the time. Figure 7 shows the Accuracy of our CNN. We got 89.55% as a result accuracy. The result is shown as a improvement of traformation, null filling and resize. Overall, Our CNN network architecture is very similar to DNN. They are combined layer by layer. The behavior between layers is also determined by the corresponding weight w and

Figure 5: The CNN architecture

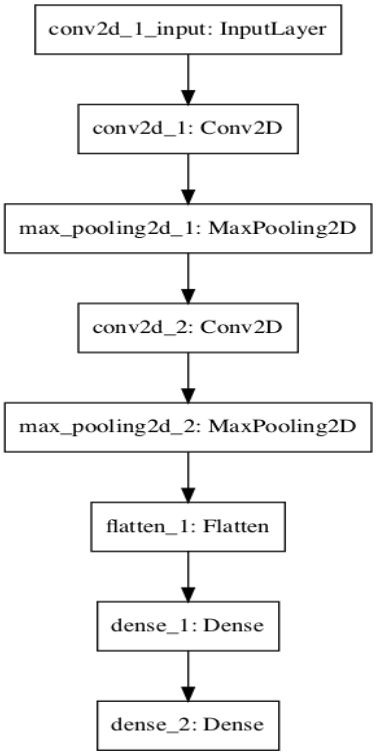


Figure 6: The Runtime Screenshot

```
model.fit(train_data.reshape(14400,150,150,1), train_label, batch_size=128, epochs=3)

WARNING:tensorflow:From /Users/charles/opt/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_back
The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

[<tf.Tensor 'conv2d_1_input:0' shape=(?, 150, 150, 1) dtype=float32>]
Epoch 1/3
14400/14400 [==============================] - 380s 26ms/step - loss: 15937716461.2849 - accuracy: 0.8899
Epoch 2/3
14400/14400 [==============================] - 378s 26ms/step - loss: 488719223794.3467 - accuracy: 0.8907
Epoch 3/3
14400/14400 [==============================] - 378s 26ms/step - loss: 2912728451062.8975 - accuracy: 0.8907
```

replacement value b, and their purpose. It is also consistent: learn w and b in the network structure through the training data, so that the input pictures can be correctly classified.

6 Discussion

In this discussion, we will talk about our report and what part we could get improvement. We test MLP model and CNN model respectively and got the best parameters with best results on Fashion MNIST dataset firstly.

Then based on the work, we permuted both datasets' training examples with the same random state and retrained and evaluated the best models of MLP and CNN. It was clear that CNN is better than MLP if data is not permuted. As in the section 4 permuted data creates a distortion on the dataset. And in this situation, MLP perform better than CNN. If the data is a noisy data MLP may be the first

Figure 7: The Screenshot of accuracy

```
score = model.evaluate(test_data.reshape(3600,150,150,1), test_label)
print('acc', score[1])

3600/3600 [==============================] - 23s 7ms/step
acc 0.8955555558204651
```

choice. From our experiments we understood that initialization of weights is too important for training process. If the right initializer has been chosen, the model can reach high accuracy in a short time. Also choosing right activation function is essential step for getting best results.

We used graphs to show the changes in the performance of the two models about loss and accuracy. We realized that GPU power strongly effect optimization of the model. In order to find best parameters we made lots of experiment, but in these experiments we had to choose low epoch rate. Otherwise either experiments take too long to compute or it cause memory error due to high computational cost.

Optimization can also be done using third party libraries. Since we wanted to see individual effects of the parameters we preferred an exhausting search method. It is possible to get better models by using other optimizer techniques.

In the final part, we used keras to implement a "Tell-the-time" network of CNN. However, we still can have more improvement in many parts. If we process the image firstly and leaving only the hour and minute hands, the model may be simpler.

7 Conclusion

To conclude, we learned keras and tensorflow and gained some experience regarding MLP an CNN networks and their optimization. There is a lot of factors which has effect on the model's performance. Hyper parameter tuning, choosing right values is mainly depends on the problem we are working on.

Task 2 also showed that not only hyper parameter tuning also the dataset we use is an important factor for evaluation. Permutation is creates an noisy dataset while keeping main features the same. We can see the performance about the two models on Permuted MNIST and Permuted Fashion MNIST. From the data comparison of our result showed in Table 5, CNN's accuracy has dropped more than MLP on Permuted MNIST. When the accuracy of CNN is declining, MLP gets some improvement on Permuted Fashion MNIST. We agreed the theory, the accuracies of MLP networks almost same as those trained on the unpermuted data and perform better than convolutional networks.

And our "Tell-the-time" network architecture is combined layer by layer and learning w and b in the network structure through the training data. That's why it can correctly classify the input pictures and the accuracy is 89.55%.

References

[1] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, 2019.